SIMULATING SENSOR NETWORKS IN NS-2 [DRAFT]

Ian Downard

Naval Research Laboratory

Code 5523

4555 Overlook Ave

Washington DC, 20375-5337

`downard@itd.nrl.navy.mil`

June 11, 2003

Abstract

Building an optimal sensor network involves deliberately addressing each of a wide range of issues, such as physically detecting phenomena, deducing environmental information from raw sensor data, and communicating important alerts through an ad-hoc wireless network to an outside observer; all under tight energy constraints. Because of the complexity of these issues, the simulation environment provides an attractive universe for sensor network engineers to investigate the consequences of using various protocols and configurations. This was the motivation behind NRL's sensor network extensions to ns-2. In this paper, we describe how we added these extensions into the ns-2 framework, and illustrate their utility with a case study examining Mobile Ad-Hoc Network (MANET) routing within a dynamic sensor network. Finally, we will describe the limitations inherent to our simulation environment.

1 Introduction

Our idea of a sensor network is an autonomous, multi-hop, wireless network with nondeterministic routes over a set of possibly heterogeneous physical layers. Routing will occur throughout the network at nodes configured in ad-hoc mode. Our purpose is to evaluate how well current routing layer standards support the requirements of various layers in these sensor networks.

The primary objective of this project was to establish a foundation in ns-2 for simulating sensor networks. This foundation, illustrated in Figure 1, consists of dual-homed sensor nodes that are tapped into an 802.11 channel for communicating with other network stations and into a phenomenon channel for detecting phenomena emanating in that medium. This work is a small contribution that should benefit sensor network research where simulation is appropriate. It is an effort to aid the analysis of various sensor network configurations under the demands of specific sensor applications.

The paper begins with an overview of ns-2, followed by a description of our extensions to ns-2 and guidelines for using them in simulations. We conclude with a section to illustrate a sensor network simulation and a final section to list important areas for future improvement of our model.

2 NS-2 Overview

The ns-2 simulation environment [1] offers great flexibility to investigate the characteristics of sensor networks because it already contains flexible models for energy constrained wireless ad-hoc networks. In the ns-2 environment, a sensor network can be built with many of the same set of protocols and characteristics as those available in the real world. The mobile networking environment in ns-2 includes support for each of the paradigms and protocols shown in Figure 2. The wireless model also includes support for node movements and energy constraints. By leveraging the existing mobile networking infrastructure, we added the capability to simulate sensor networks. The only fundamental aspect of sensor networks missing in ns-2 was the notion of a phenomenon such as chemical clouds or moving vehicles that could trigger nearby sensors through a channel such as air quality or ground vibrations. Once a sensor detects the "ping" of a phenomenon in that channel, the sensor acts according to the sensor application defined by the ns-2 user. This application defines how a sensor will react once it detects its target phenomenon. For example, a sensor may periodically send a report to some data collection point as long as it continues to detect the phenomenon, or it may do something more sophisticated, such as collaborate with neighboring sensor nodes to more accurately characterize the phenomenon before alerting any outside observer of a supposed occurrance. For each sensor network there is a unique sensor application to accomplish phenomena detection, such as surveillance, environmental monitoring, etc. With ns-2, we have provided the facility to invoke sensor applications by phenomena. With these sensor applications, we can study how the underlying network infrastructure performs under various constraints.

3 Related Work

Modeling sensor networks for simulation has not received as much research as other areas like information processing and energy conservation. In fact, we are aware of only one project whose objective included building flexible simulation tools specifically for sensor networks. Park, Savvides, Srivastava [7] developed extensions to ns-2 for this purpose, but with an emphasis on sophisticated modeling of energy consumption and emulation (i.e. interfacing with real world sensor nodes). Unfortunately, their work has not been updated to support subsequent releases of ns-2 since October, 2000.

4 The Extended NS-2 Architecture

We modeled the presence of phenomena in ns-2 with broadcast packets transmitted through a designated channel. The range of phenomena is the set of nodes that can receive the PHENOM broadcast packets in that channel[1]. This pattern will follow whichever radio propagation model (free space, two ray ground, or shadowing) included with the phenomenon node's configuration. These propagation models roughly cover a circle, but other shapes could be achieved by varying the range of PHENOM broadcast packets and creatively moving a set of phenomenon nodes emanating the same type of phenomenon.

---

[1]This reflects the range of sensitivity of the sensors. For example, PHENOM broadcasts with a long range would simulate highly sensitive sensors. The sensitivity of a single sensor can be controlled by setting the receive and carrier sense thresholds in defined in `mac/wireless-phy.cc`.
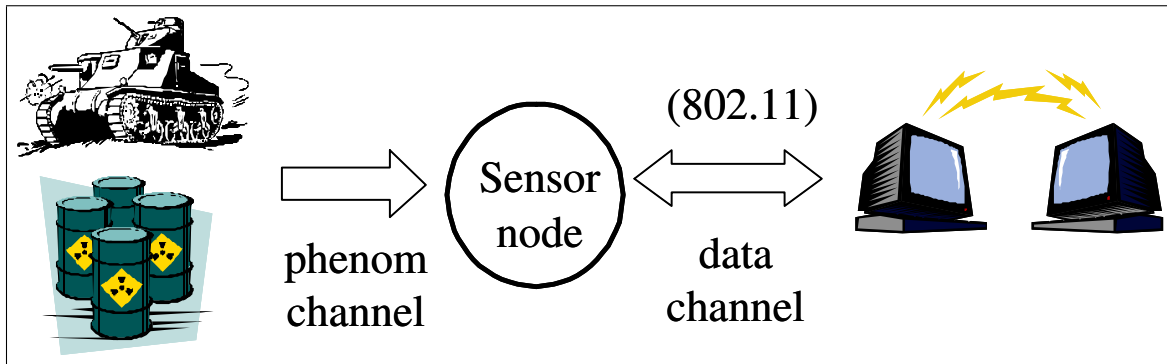
Figure 1: This is the foundation of the sensor network model used in ns-2.
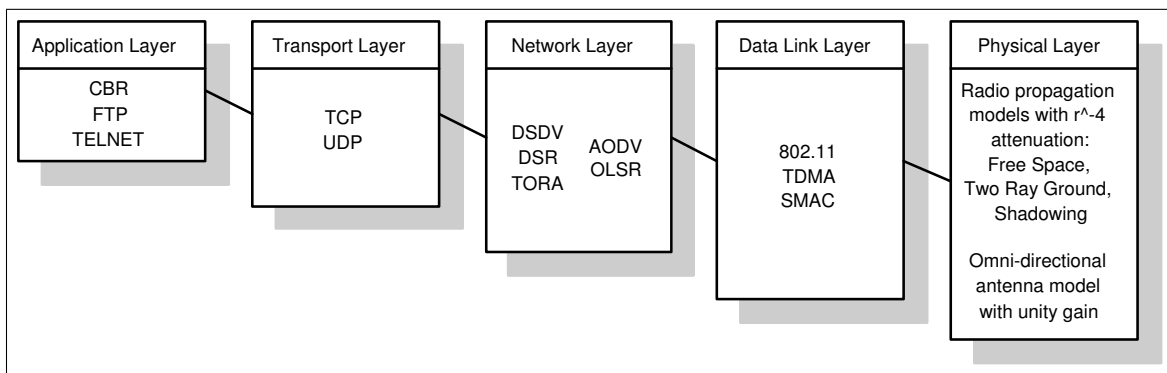

Figure 2: These are some of the paradigms and protocols available for wireless networking in ns-2. Some protocols like OLSR [4] and SMAC [6] have not yet been incorporated into USC's ns-2 distributions [1], but they can be retrieved from their respective developers' sites.

Emanating PHENOM broadcast packets is accomplished by the "PHENOM routing protocol"[2], which simply broadcasts PHENOM packets with a certain configurable pulserate. When a PHENOM packet is received by a node listening on the phenomenon channel, a receive event is passed to that node's sensor application.

4.1 Modifications to NS-2

Figure 3 shows where our extensions are arranged within the ns-2 framework. The major additions and modifications are explained below. Section 4.3 shows how our extensions fit into ns-2's class hierarchy.
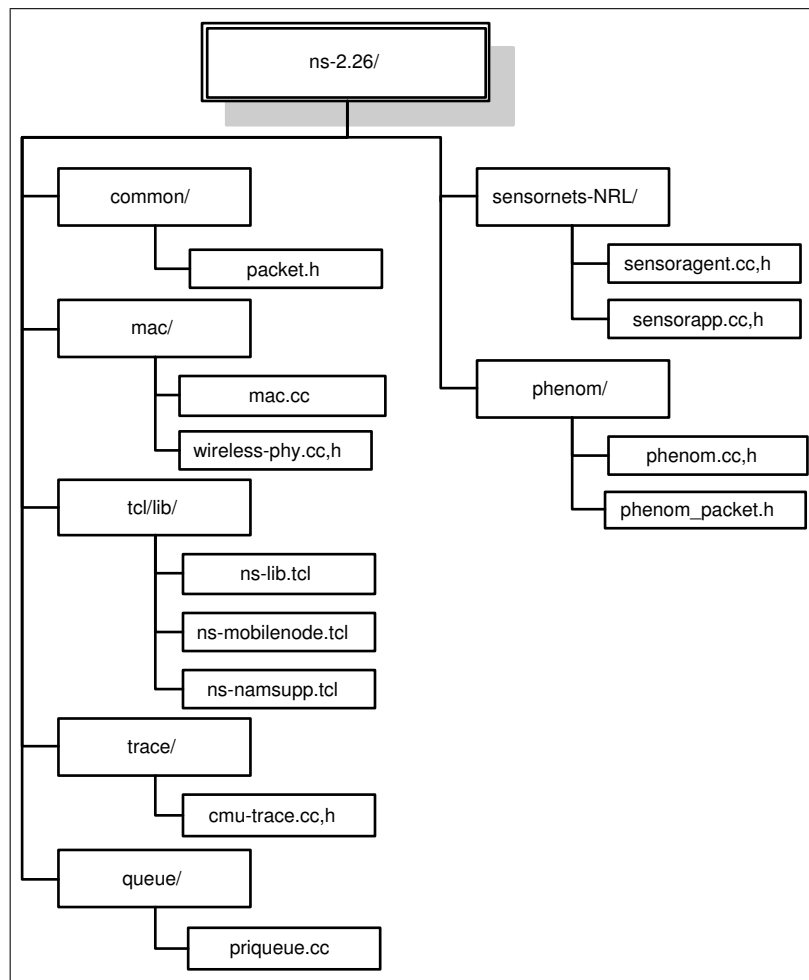

Figure 3: This figure illustrates which files in the ns-2 framework were modified (see left side) or added (see right side).

---

[2]This functionality best fit into ns-2's existing ad-hoc wireless networking infrastructure as a routing protocol, even though it does not actually route at all. The MAC layer it operates above must be specified in the phenomenon node's configuration. Although real-world phenomena can interfere in a variety of ways, we ignore this aspect and use the basic "Mac" class, which seems to prevent channel contention.

`trace/cmu-trace.cc,h` The `CMUTrace` class is used to print important parts of a packet to the simulation's trace file. Since we introduced a new packet type for phenomena, we had to describe the corresponding packet format in this class.

`tcl/lib/ns-lib.tcl` This component of the infrastructure interprets node configurations specified in the ns-2 simulation script. Our extensions introduced two new node types, the sensor node and the phenomenon node. Therefore, we added some arguments in the `node-config` function to accommodate them.

`tcl/lib/ns-mobilenode.tcl` In ns-2's virtual world, we're using its existing capacity for multichannel wireless networking as a means to emanate phenomena of various kinds. By using a dedicated channel for phenomena, we can simulate the unique physical medium that they occupy in the real world. Thus, as shown in Figure 1, sensor nodes will need to have two interfaces, one to the 802.11 channel and one to the PHENOM channel. We implemented this kind of "multi-homed" capability in `ns-mobilnode.tcl`.

`common/packet.h` Each packet in ns-2 is associated with a unique type that associates it with the protocol that it belongs to, such as TCP, ARP, AODV, FTP, etc. Since we created a new protocol for emanating phenomena, we defined it's corresponding packet type in the `packet.h` header file.

`mac/wireless-phy.cc` Ns-2 contains an energy model for wireless nodes that can be used to investigate the benefits of various energy conservation techniques, such as node sleeping or utilizing optimal network densities. The model includes attributes for specifying the power requirements of transmitting packets, receiving packets, or idly standing by during times of network inactivity. Sensing phenomena is a process that may consume power at another rate, so it's important to consider this where sensor network simulations are concerned. In `mac/wireless-phy.cc`, we've included the capability of specifying the amount of power consumed by nodes while sensing phenomena.

Other small modifications were made to `mac/mac.cc`, `tcl/lib/ns-namsupp.tcl`, and `queue/priqueue` in order to facilitate the second interface to the phenomenon channel on sensor nodes, to fix a bug in ns-2's node coloring procedure, and to include the new PHENOM packet type into the ns-2 framework.

4.2 Additions to NS-2

Every sensor network simulation must have phenomenon nodes that trigger sensor nodes, but the traffic sensor nodes generate once they detect phenomena depends on the function of the sensor network. For example, sensor networks designed for energy efficient target tracking [9] would generate more sensor-to-sensor traffic than a sensor network designed to provide an outside observer with raw sensor data. This function is defined by the sensor application which is intended to be customized according to the traffic properties associated with the sensor network being simulated. The objects and functions we have just described are implemented in the following files:

`phenom/phenom.cc,h` This file implements the PHENOM routing protocol used for emanating phenomena. It includes parameters for the pulse rate and the phenomenon type (Carbon Monoxide, heavy seismic activity, light seismic activity, sound, or generic). These types are just names that can

be used to identify multiple sources of phenomena in trace files. The pulse rate is the only parameter that actually controls how a phenomenon emanates.

`sensornets-NRL/sensoragent.cc,h` The ns manual [2] describes *agents* as "endpoints where network-layer packets are constructed or consumed". Sensor nodes use a *sensor agent* attached to the phenomenon channel for consuming PHENOM packets, and a UDP or TCP agent attached to the wireless network channel for constructing packets sent down from the sensor application. Sensor agents act as a conduit through which PHENOM packets are received and processed by sensor applications. The sensor agent does not actually look at the contents of the PHENOM packet, it simply marks the packet as received and passes it to the sensor application. This agent is implemented in `sensoragent.cc`.

`sensornets-NRL/sensorapp.cc,h` The sensor application defined in this file utilizes node color and generates sensor reports to show when the corresponding sensor node detects phenomenon[3]. Specifically, when the node is receiving PHENOM packets, this application changes the node color to red, activates an "alarm" public variable, and sends a sensor report of MESG_SIZE bytes to the sink node of a UDP (or TCP) connection once per TRANSMIT_FREQ seconds. When the node has not received a PHENOM packet in the timeout period specified by SILENT_PHENOMENON, then the node color changes back to green. If node color is desired to illustrate energy levels instead of sensor alarm status, then that aspect of the application can be disabled with DISABLE_COLORS.

A visualization of this sensor application is shown in Figure 4.

`sensornets-NRL/phenom_packet.h` This file defines the structure of PHENOM packets. The five phenomenon types defined here (CO, HEAVY_GEO, LIGHT_GEO, SOUND, and TEST_PHENOMENON) correspond to Carbon Monoxide, heavy seismic activity, light seismic activity, audible sound, and some generic phenomenon. These types are most useful for simulations involving multiple phenomenon nodes, in order to easily distinguish who a given sensor node is detecting by looking at the ns-2 trace file.

4.3 The Extended NS-2 Class Hierarchy

The Doxygen documentation system [8] was used to generate Figures 5, 6, and 7 that illustrate how our extensions fit into ns-2's class hierarchy. Dotted lines show where a class is using the methods and members of another class. Solid lines show where a class is inheriting the methods and members from another class.

5   Capabilities, Guidelines, and Caveats.

This section describes the capabilities of our sensor network extensions, gives some guidelines for configuring simulations, and attempts to explain some areas of likely confusion. In this section, we assume the reader is already familiar with setting up mobile node simulations in ns-2. For readers who are not, the following URLs provide background:

---

[3]The four environment variables that can be used to customize this application are SILENT_PHENOMENON, DISABLE_COLORS, MESG_SIZE, and TRANSMIT_FREQ.
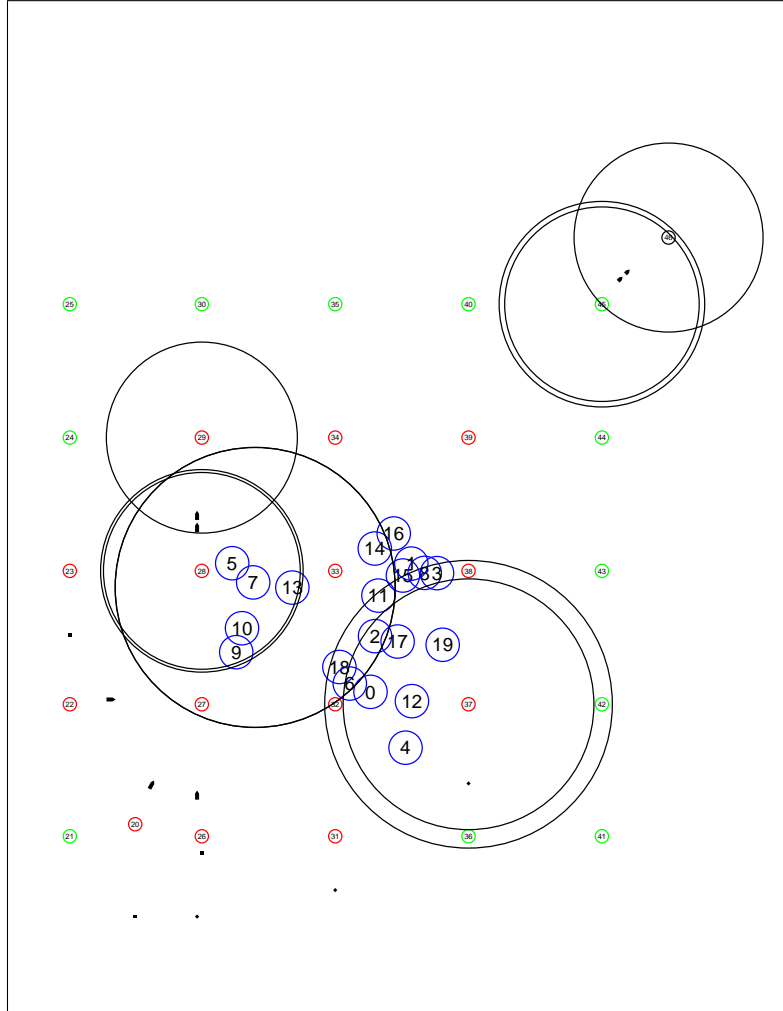
Figure 4: Visualization of a simulated sensor network with 25 stationary sensor nodes, 20 mobile phenom nodes simulating a gas cloud, and one stationary data collection point. The red sensor nodes detect the phenomenon, the green ones do not. The phenomenon nodes are large and blue, and the data collection point is the black node in the far upper-right corner.
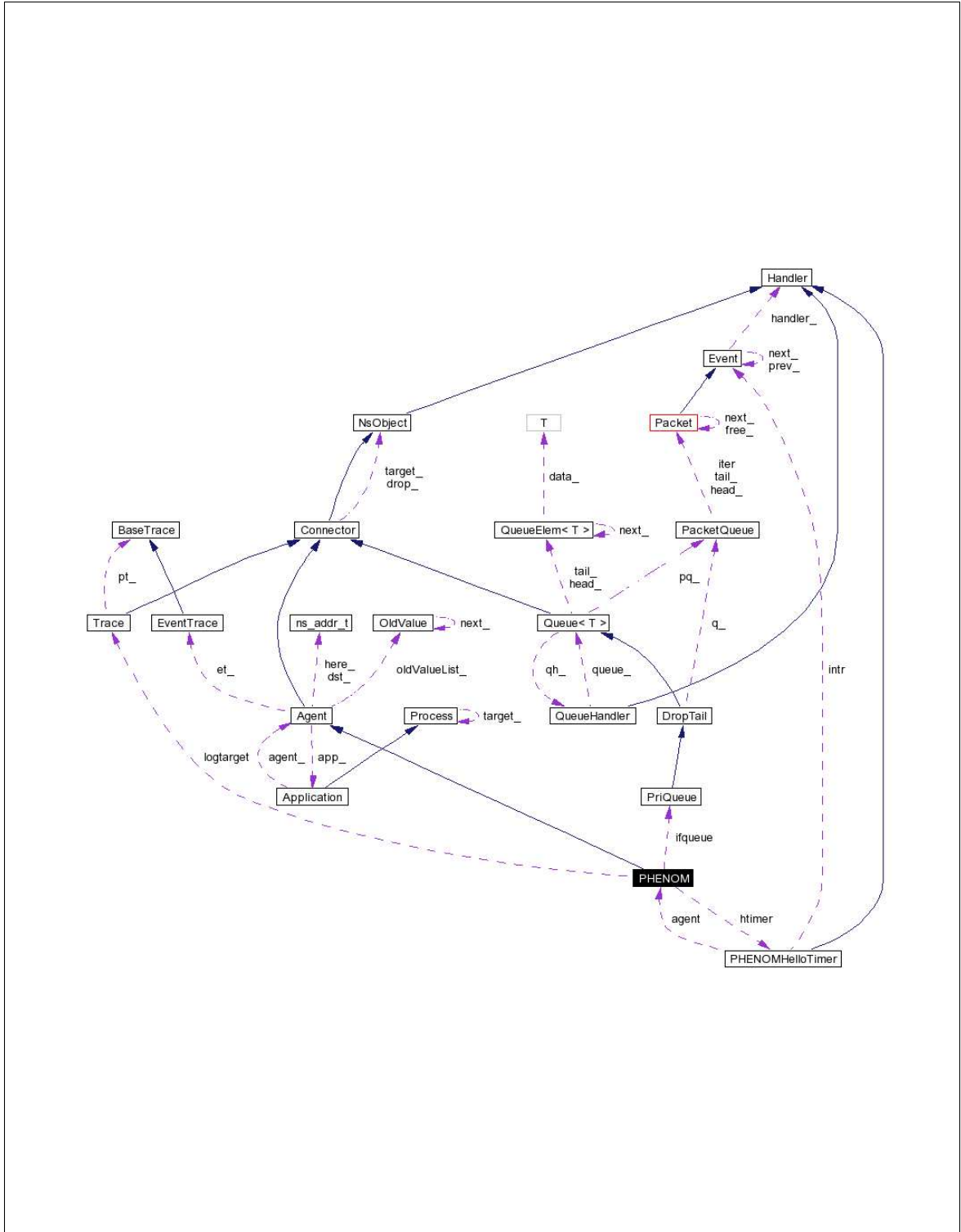
Figure 5: Collaboration diagram for the PHENOM class.

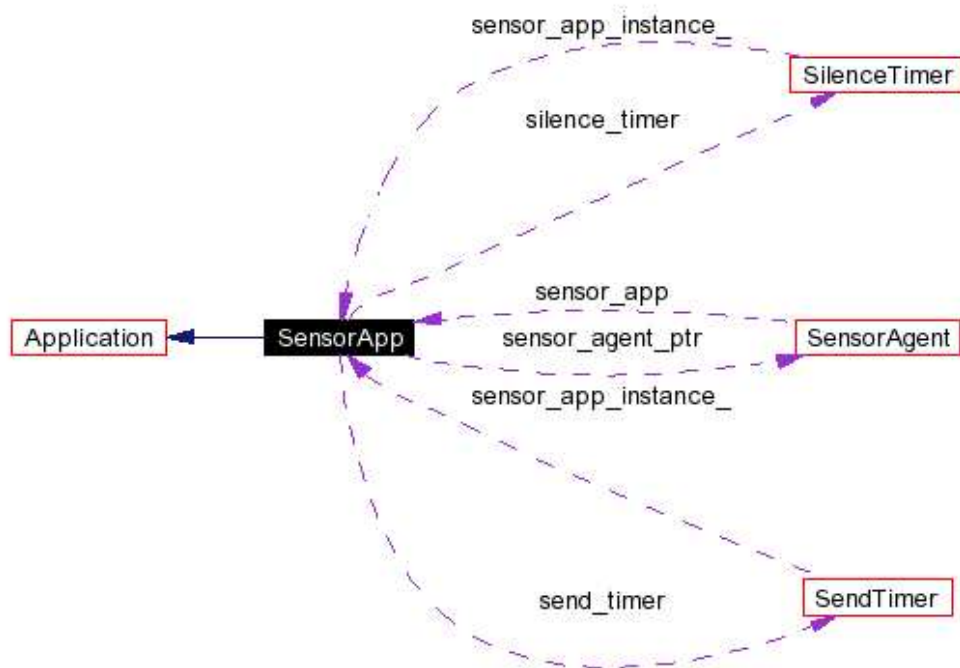Figure 6: Collaboration diagram for the SensorAgent class.

Figure 7: Collaboration diagram for the SensorApp class.

```
http://nile.wpi.edu/NS/
http://www.isi.edu/nsnam/ns/tutorial/index.html
http://www.isi.edu/nsnam/ns/tutorial/nsscript5.html
```

The easiest way to create sensor network simulations is to use the `script_maker.pl` utility in the `simulations_aids` directory distributed with our extensions. This Perl script contains commonly used parameters for setting up sensor network simulations and automatically generates the often complex ns simulation script. The remainder of this section describes how to code a sensor network simulation into the ns simulation script, without using the `script_maker.pl` utility.

Setting up a sensor network in ns-2 follows the same format as mobile node simulations. The best way to create your own simulation is to modify one of the examples distributed with our code [3].

Places where a sensor network simulation differs from a traditional mobile node simulation are listed below. Setting up `ns_`, `god_`, tracing, topography objects and starting and stopping the simulation are all the same as in traditional mobile node simulations.

1. Configure a phenomenon channel and data channel.

   Phenomenon nodes should emenate in a different channel than sensor nodes in order to avoid contention at the physical layer. All phenomenon nodes should be configured on the same channel, even if they're emanating different types of phenomena.

   ```
   set chan_1_ [new $val(chan)]
   set chan_2_ [new $val(chan)]
   ```

2. Configure a MAC protocol for the phenomenon channel.

   Choose a MAC layer to use for emanating phenomena over the phenomenon channel. Using 802.11 probably isn't appropriate, since phenomena should be emanating without regard to collisions or congestion control. We suggest using the basic "Mac" class instead.

   ```
   set val(mac)        Mac/802_11  ;# MAC type for sensor nodes
   set val(PHENOMmac)  Mac         ;# MAC type for phenomena
   ```

3. Configure phenomenon nodes with the PHENOM "routing" protocol:

   Use node-config, just like with mobile nodes, but specify PHENOM as the routing protocol so the phenomenon is emanated according to the methods defined in `phenom/phenom.cc`. Also, be sure to configure in the channel and MAC layer previously specified for phenomena broadcasts.

   ```
   $ns_ node-config \
       -adhocRouting PHENOM \
       -channel $chan_1_ \
       -llType LL \
       -macType $val(PHENOMmac) \
       -ifqType Queue/DropTail/PriQueue \
       -ifqLen 50 \
   ```

```
-antType Antenna/OmniAntenna \
-propType Propagation/TwoRayGround \
-phyType Phy/WirelessPhy \
-topoInstance $topo \
-agentTrace ON \
-routerTrace ON \
-macTrace ON \
-movementTrace ON
```

4. Configure the Phenomenon node's pulse rate and type.

   The two parameters that can be used to customize phenomena are listed below. They are both optional.

   (a) `pulserate FLOAT`
      - `FLOAT` must be a real number.
      - Describes how frequently a phenomenon node broadcasts its presence.
      - Defaults to 1 broadcast per second.

   (b) `phenomenon PATTERN`
      - `PATTERN` must be any one of the following keywords: CO, HEAVY_GEO, LIGHT_GEO, SOUND, TEST_PHENOMENON corresponding to Carbon Monoxide, heavy seismic activity, light seismic activity, audible sound, and some other generic phenomenon.
      - This option is mostly useful for simulations involving multiple phenomenon nodes, so that it is easier to distinguish who a sensor node is detecting by looking at the NS trace file.
      - Defaults to TEST_PHENOMENON.

   ```
   [$node_(0) set ragent_] pulserate .1 ;#PHENOM emanates 10x/s
   [$node_(0) set ragent_] phenomenon CO ;#Carbon Monoxide PHENOM
   ```

5. Configure sensor nodes.

   Sensor nodes must be configured with the `-PHENOMchannel` attribute and the `-channel` attribute. `PHENOMchannel` must be the same as the channel you configured the phenomenon node with. The other channel is the channel that will be used for communicating sensor reports. Sensor node configurations must also specify a MAC protocol for the phenomena channel and a MAC protocol (such as `Mac/802_11`) for the channel shared with other wireless nodes. This is done with the `-PHENOMmacType` and `-macType` attributes. `PHENOMmacType` should be the same as the macType used in PHENOM nodes, and `macType` should be the same as the macType used in other nodes participating in the MANET network (such as 802.11).

   ```
   $ns_ node-config \
      -adhocRouting $val(rp) \
      -channel $chan_2_ \
      -macType $val(mac) \
      -PHENOMmacType $val(PHENOMmac) \
      -PHENOMchannel $chan_1_
   ```

If desired, a sensor node can be configured so that a specified amount of energy will be deducted from its energy reserve each time it receives a phenomenon broadcast. To set this up, include the following parameters in the sensor node's `node-config` routine:

```
-energyModel EnergyModel \
-rxPower 0.175 \
-txPower 0.175 \
-sensePower 0.00000175; \
-idlePower 0.0 \
-initialEnergy 0.5
```

where,

`rxPower` `.175` indicates $175mW$ consumed for receiving a packet of arbitrary size

`txPower` `.175` indicates $175mW$ consumed for transmitting a packet of arbitrary size

`sensePower` `.00000175` indicates $1.75\mu W$ consumed for detecting receiving a PHENOM broadcast packet

`initialEnergy` `5` indicates a total energy reserve of $5J$ available to the sensor

IMPORTANT CAVEAT:

Ns-2's energy consumption model utilizes color to illustrate when a node is about to exhaust it's energy. In order to avoid confusion in the nam visualization, the node coloring that is part of the sensor application should be disabled with the DISABLE_COLORS definition in `sensorapp.cc`. Remember to run make again to compile those changes into the ns-2 executable.

In addition to DISABLE_COLORS, some other sensor node parameters can be specified in `sensorapp.cc`. These parameters are listed below:

SILENT_PHENOMENON is the seconds of quiescence required for a sensor to go off it's alarming state. Example:

> `#define SILENT_PHENOMENON 0.2`

DISABLE_COLORS disables node color changes invoked by the sensor application. This is useful when it is desired to use node color to illustrate a node's energy reserves. Example:

> `#define DISABLE_COLORS FALSE`

MESG_SIZE is the size (in bytes) of the messages to send to the gateway, or data collection point, or whatever you want to call the sink node attached to this sensor node (over UDP, for example). Example:

> `#define MESG_SIZE 256`

TRANSMIT_FREQ is the frequency with which a sensor node triggered by PHENOM packets will send a message to the the sink node attached to this sensor node. Units are in seconds, so a message of size MESG_SIZE bytes will be transmitted to the gateway node once for every TRANSMIT_FREQ seconds in which the sensor node has received one or more PHENOM packets.

> `#define TRANSMIT_FREQ 0.1`

6. Configure non-Sensor nodes (such as data collection points, or gateways for the sensor network).

   Nodes that are not sensor nodes or phenomenon nodes, should not be configured with a `PHENOM-channel`, since their only interface is to the MANET network. This is done with the `-PHENOMchannel "off"` attribute.

```
$ns_ node-config \
    -adhocRouting $val(rp) \
    -channel $chan_2_ \
    -PHENOMchannel "off"
```

7. Attach sensor agents.

   Create a sensor agent for each sensor node, and attach that agent to its respective node. Also, specify that all packets coming in from the PHENOM channel should be received by the sensor agent. In the following example, `$i` would represent the node number for the sensor node currently being configured.

```
set sensor_($i) [new Agent/SensorAgent]
$ns_ attach-agent $node_($i) $sensor_($i)

# specify the sensor agent as the up-target for the sensor node's link
# layer configured on the PHENOM interface, so that the sensor agent
# handles the received PHENOM packets instead of any other agent
# attached to the node.
[$node_($i) set ll_(1)] up-target $sensor_($i)
```

8. Attach a UDP agent and sensor application to each node (optional).

   How the sensor nodes react once they detect their target phenomenon is a behavior that should be defined as a sensor application. One such application might involve sensor nodes alerting a data collection point via UDP with information about the phenomenon. The following example illustrates how an application like that could be setup. Again, `$i` represents the node number for the sensor node currently being configured.

```
set src_($i) [new Agent/UDP]
$ns_ attach-agent $node_($i) $src_($i)
$ns_ connect $src_($i) $sink

set app_($i) [new Application/SensorApp]
$app_($i) attach-agent $src_($i)
```

9. Start the sensor application.

   The sensor node will receive PHENOM packets[4] as soon as the sensor agent is attached to the node. Since the sensor agent does nothing but notify the sensor application of received phenomenon broad-

---

[4]Phenomenon nodes start emanating immediately once the simulation starts. A delayed start can be realized by reducing the range of phenomenon broadcasts to such a small area that they are effectively inaudible to any sensors (unless they occupy the exact same coordinate in the grid). A phenomenon node can be turned off with a command like, `$ns_ at 6.0 {[$node_($i) set netif_(0)] set Pt_ 0.0001}`. `Pt_` is the range of the broadcast, and `$i` is the node id of the Phenomenon node.

casts, the sensor node does not visibly react to PHENOM packets until the sensor application has been attached and started. The following example shows how to start a sensor application:

```
$ns_ at 5.0 "$app_($i) start $sensor_($i)"
```

6 Case Study: MANET Routing Within a Dynamic Sensor Network

This case study begins to show the types of results one can achieve from sensor network simulations with ns-2. Suppose we'd like to characterize how well AODV scales with the size of a sensor network running the sensor application defined at the end of section 4. We will look at networks of stationary sensors with infinite energy placed in a grid with d units of distance between adjacent nodes. Our network will range from 50 sensor nodes to 2000 sensor nodes. We will limit the broadcast range of 802.11 radios and the range of the phenomenon to $\sqrt{2}d$, as shown in Figure 8. Since we're using the Two-ray Ground radio propagation model, nodes within this boundary always receive the broadcast and nodes outside never receive the broadcast.[5]
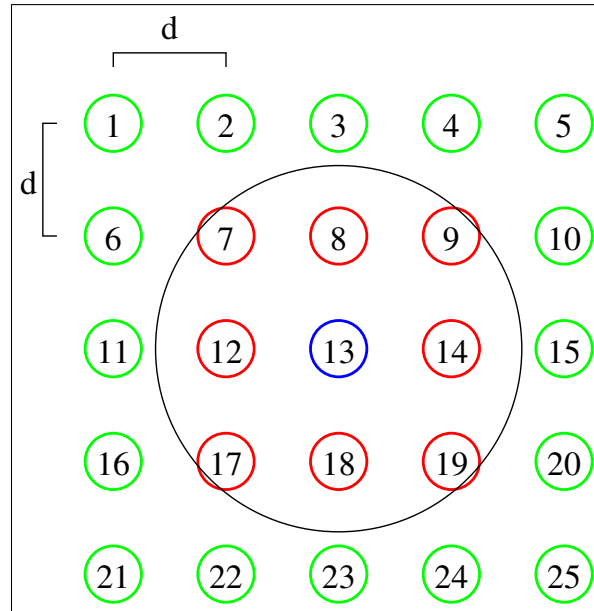

Figure 8: This figure illustrates the maximum broadcast used in our case study. If we use the Two-ray Ground radio propagation model, then the broadcast range for node 13 can never broadcast further than the ideal circle with radius $\sqrt{2}d$.

We will excite the network with a single phenomenon node that slowly travels along the perimeter of the sensor network grid. As the grid density increases, the phenomenon will encounter sensor nodes more frequently. Thus, as the grid density increases, AODV will flood more route requests through the network. As the network becomes more congested, we should observe higher latency and higher loss rates in sensor

---

[5]In reality, this boundary is a random variable due to complex fading and interference effects.
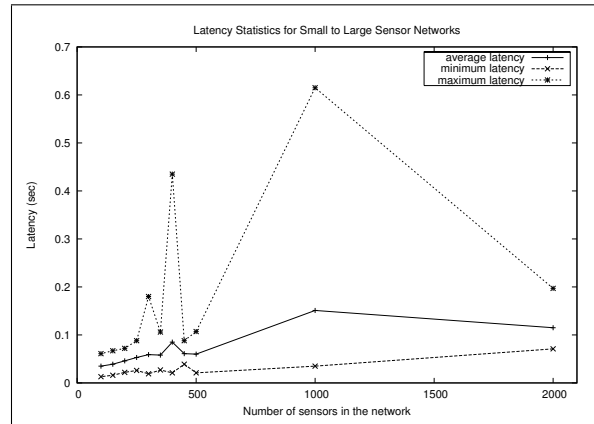
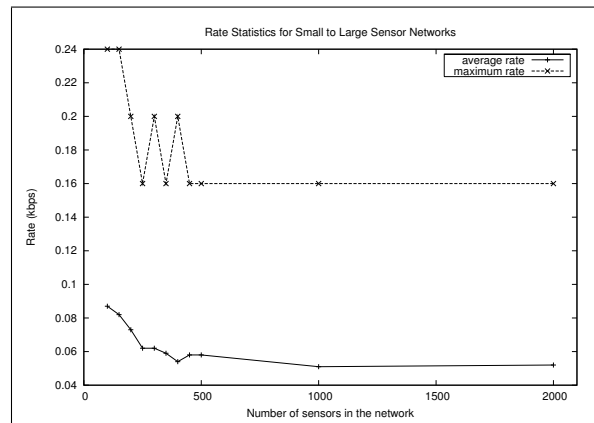Figure 9: Latency shown as a function of network size.

Figure 10: Data rates shown as a function of network size.

reports delivered to the stationary data collection point. See Figures 9, 10, and 11 for latency, data rate, and loss fraction statistics.

THIS SECTION IS INCOMPLETE

- What's the point of this section?
  (To show the utility in our ns-2 extensions).

- How does this case-study proove that point?
  (By illustrating complex traffic patterns which would be a bear to reproduce without said ns-2 extensions).

- How did we generate the plots?
  (With MGEN.)

- What is MGEN?
  (Oh great! Don't *you* start...)

7 Software References

7.1 NRL Sensor Network Extensions to NS-2

NRL's code and documentation for extending ns–2.1b9a or ns–2.26 to support sensor network simulations is available here:

```
http://pf.itd.nrl.navy.mil/projects/nrlsensorsim/
```

7.2 NRL OLSR Extensions to NS-2

One of the original motivations behind building the sensor network extensions into ns-2 was to compare the behaviors of OLSR and AODV routing algorithms. NRL's OLSR extension to NS–2 is available for download via CVS [2]. To check out all the nrlolsr files (from a Linux box), type the following two commands:

```
cvs -d :pserver:anonymous@nrlolsr.pf.itd.nrl.navy.mil:/cvsroot/nrlolsr login
cvs -z3 -d :pserver:anonymous@nrlolsr.pf.itd.nrl.navy.mil:/cvsroot/nrlolsr co .
```

The instructions for including nrlolsr extensions in ns-2 are documented in nrlolsr/readme. To get the code, follow the nrlolsr-NS- link at this page:

```
http://pf.itd.nrl.navy.mil/projects/olsr/
```

8 Bugs

Phenomenon nodes receive broadcasts from other phenomenon nodes. This doesn't seem to effect simulation results on the IP side of the network, but it does make the simulations much longer and trace files much larger when multiple phenomenon nodes are being used in close proximity.

Please direct all bug reports to Ian Downard, `<downard@itd.nrl.navy.mil>`.

9 Future Work

Much more effort should be made to improve how phenomenon emanates. Presently, it follows the behavior of an 802.11 broadcast, configured with one of the following radio propagation models:

1. Free Space Model

2. Two Ray Ground Model

3. Shadowing Model

The first two models represent the communication range as an ideal circle, whose boundary is an absolute limit on signal range. The Shadowing model applies a more probabilistic means of determining whether a receiver on the boundary can receive the signal.

Using a radio propagation model to simulate anything other than electromagnetic wave propagation is probably unrealistic. So, the radio propagation model should be extended to create various phenomenon propagation models that could specifically address the characteristics of phenomena such as seismic wave propagation and gas dispersion.

10 Conclusion

The primary contribution of our extensions to ns-2 is the capability to invoke network traffic in manners consistent to the patterns expected for sensor networks. Our notion of sensor applications responding to a phenomenon node moving through a grid of sensor nodes is analogous to the Frisbee model [10], where the set of active sensors follows under the range of a mobile phenomenon. Coordinating these unique traffic patterns in ns-2 without our extensions requires very much effort for medium to large networks. Aside from generally increasing the flexibility of ns-2, this work facilitates our objective to evaluate how well current MANET routing protocols support the requirements of various sensor network applications.

References

[1] The Network Simulator - ns-2, `http://www.isi.edu/nsnam/ns/`

[2] The ns Manual, `http://www.isi.edu/nsnam/ns/ns-documentation.html`

[3] NRL's Sensor Network Extension to ns-2, `http://nrlsensorsim.pf.itd.nrl.navy.mil/`

[4] NRL's OLSR implementation for ns-2, `http://pf.itd.nrl.navy.mil/projects/olsr/`

[5] Josh Broch, David A. Maltz, David B. Johnson, Yih-Chun Hu, Jorjeta Jetcheva. "A performance of Multi-Hop Wireless Ad Hoc Network Routing Protocols," in proceedings of the Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM), 1998.

[6] Wei Ye, John Heidemann, Deborah Estrin. "An Energy-Efficient MAC Protocol for Wireless Sensor Networks," in proceedings of the IEEE INFOCOM, 2002.

[7] SensorSim: A Simulation Framework for Sensor Networks. `http://nesl.ee.ucla.edu/projects/sensorsim/`

[8] The Doxygen documentation system, `http://www.doxygen.org`

[9] H. Yang, B. Sikdar. "A Protocol for Tracking Mobile Targets using Sensor Networks," `Proceedings of the First IEEE International Workshop on Sensor Network Protocols and Applications`, pp. 71-81, Anchorage, AK, May 2003.

[10] A. Cerpa, J. Elson, D. Estrin, L. Girod, M. Hamilton, and J. Zhao, "Habitat Monitoring: Application Driver for Wireless Communica-tions Technology," `Proceedings ACM SIGCOMM Workshop on Data Communications in Latin America and the Caribbean,` April 2001.
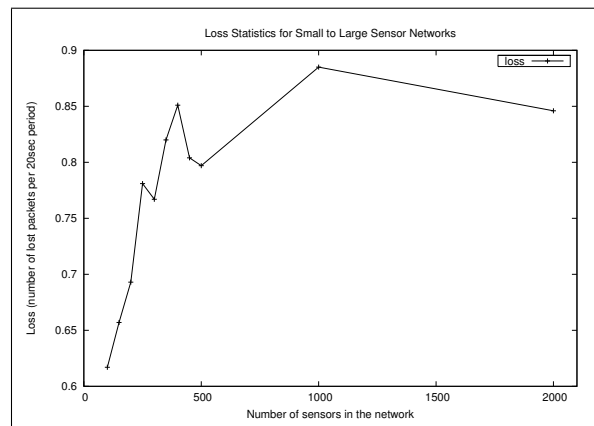
Figure 11: Loss fractions shown as a function of network size.